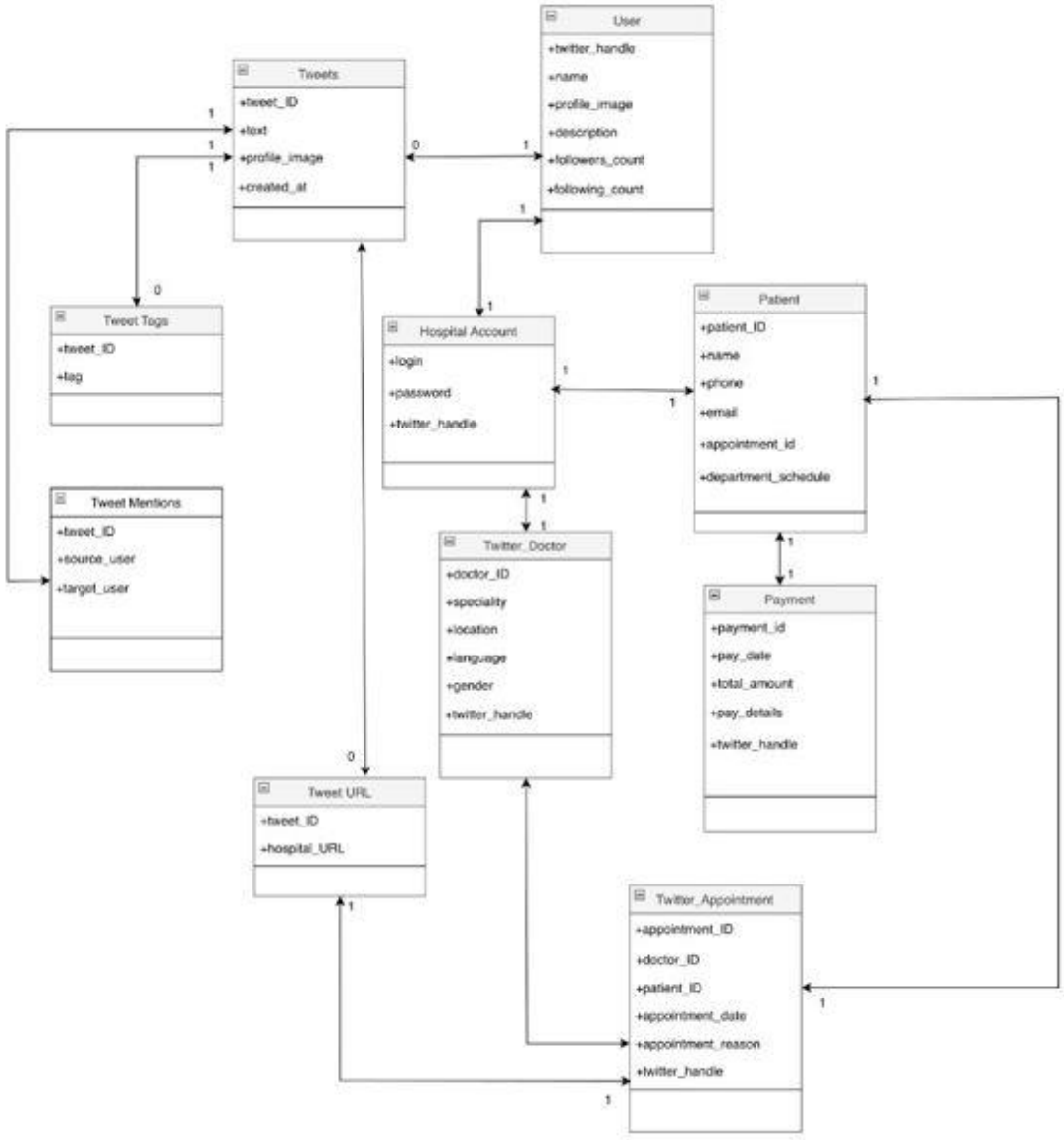
DMDD ASSIGNMENT- 2

Anuja Shinde (002747948)

Anuja Kale (002700699)

Entity Relationship Diagram:

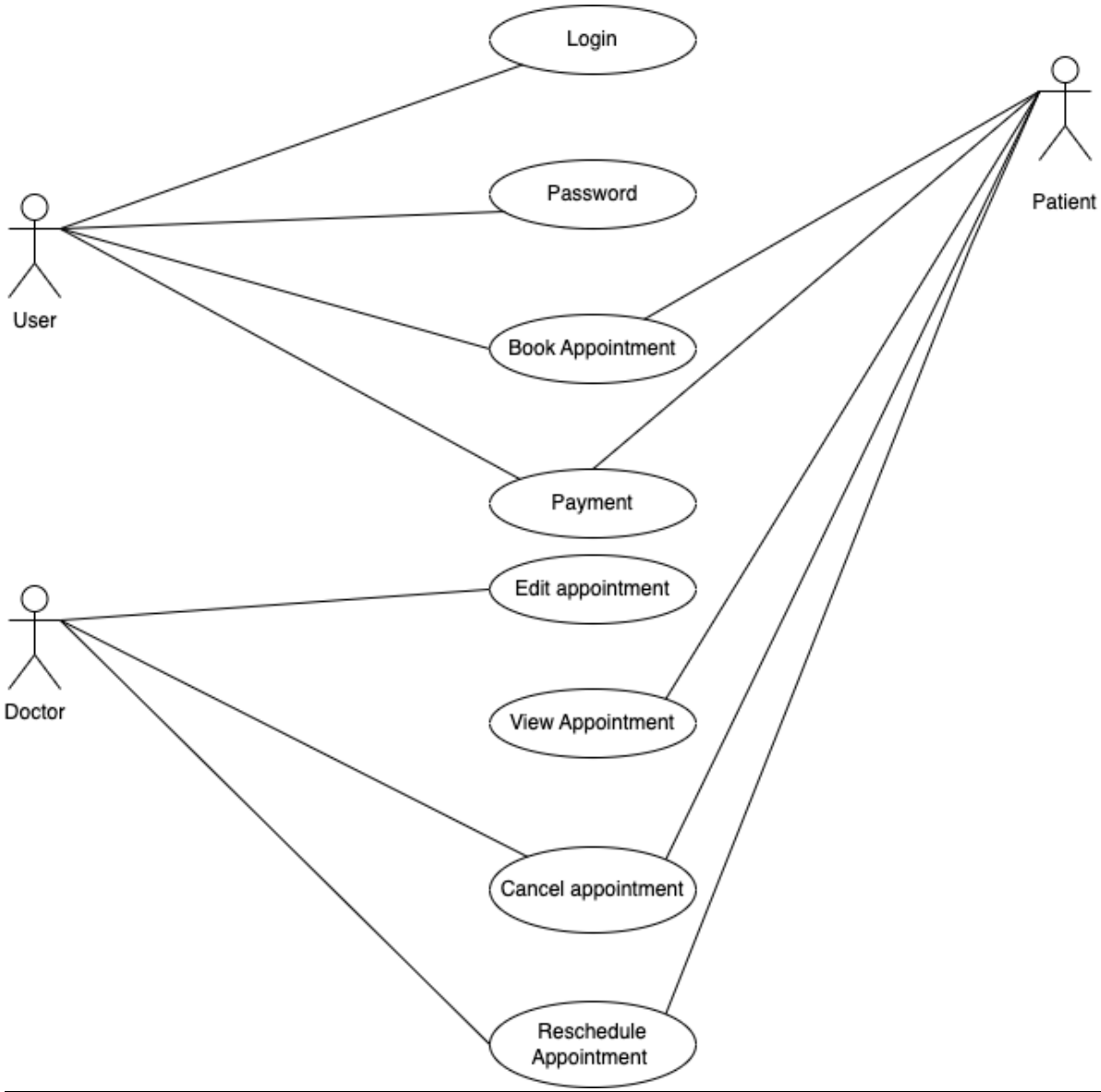


Explanation of some of the design decisions:

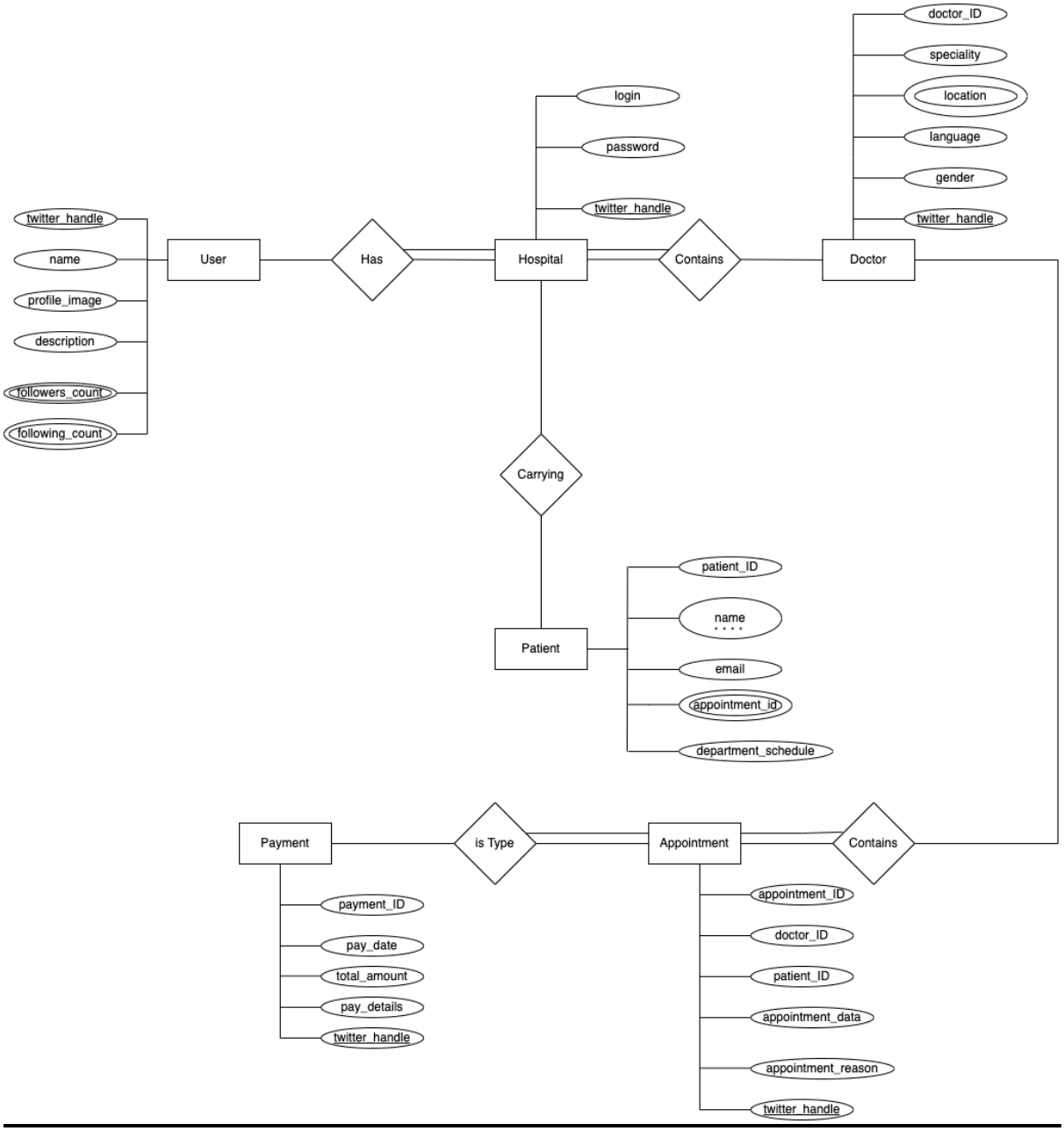
- The Brigham and Women's Hospital account has a login and password. This login is the same as a user's Twitter handle. The Twitter handle is unique – hence it can also be treated as the primary key of the table.
- Each user can tweet any number of tweets. The Brigham and Women's Hospital - user admin user of the Brigham and Women's Hospital is also one of the users and this information can be stored in the patient table itself.

- A patient can make an appointment through Twitter by tweeting the appointment detail and mentioning the doctor's account URL.
- ‘Payment’ has the ‘payment_id’ of the tweet which uniquely distinguishes each tweet, ‘appointment-url’ which is a foreign key reference to the ‘appointment_url’ in the ‘Tweet_Url’ table, ‘appointment_id’ which is a foreign key reference to ‘appointment_id’ in ‘Women_Appointments’ table corresponding to the particular ‘appointment_url’ mentioned by a patient in the tweet.
- A patient can tweet how many ever appointments he/she wants. Hence the ‘payment’ has a ‘payment_id’ which is the primary key of the table (since it uniquely distinguishes each order). Note that each appointment in the payment can have more than one Twitter appointment.

Use Case Diagram:



UML Diagram:


doctor_ID
speciality
location
language
gender
twitter_handle
login
password
twitter_handle
twitter_handle
name
profile_image
description
followers_count
following_count
User
Has
Hospital
Contains
Doctor
Carrying
patient_ID
name
email
appointment_id
department_schedule
Patient
Payment
is Type
Appointment
Contains
payment_ID
pay_date
total_amount
pay_details
twitter_handle
appointment_ID
doctor_ID
patient_ID
appointment_data
appointment_reason
twitter_handle

Use Cases:

1. Use Case: Register for an account at Brigham and Women's Hospital
Description: The user registers for an account in Brigham and Women's Hospital
Actor: User
Precondition: When a patient wants to book an appointment in the hospital, firstly he will be registered
Steps:
Actor action: User request for registration
System Responses: If patient information is correct then the patient is registered and the use case ends.
Post Condition: Patient successfully registered
Alternate Path: The customer request is not valid and the system throws an error
Error: User information is incorrect

2. Use Case: Make an appointment at Brigham and Women's Hospital
Description: The patient makes an appointment at Brigham and Women's Hospital
Actors: Patient
Precondition: The patient must have a unique Twitter handle to tweet
Steps:
Actor action – Patient tweets about an appointment along with the doctor's URL
System Responses – An appointment is made for the patient that matches the appointment reason with respect to that particular qualified doctor's URL
Post Condition: An appointment is added to the department_schedule table for the appointment the user tweeted.
Alternate Path: The appointment is not currently available in the Brigham and Women's Hospital
Error: Appointment Not Available

3. Use Case: View an appointment already booked through Twitter by a patient
Description: The patient views an appointment already booked
Actors: Patient
Precondition: The patient must have made an appointment
Steps:
Actor action – The patient views an appointment from its URL
System Responses – appointment URL would be displayed
Post Condition: system displays the appointment URL

4. **Use Case:** View the appointments above a particular price (say $100)
Description: Use view the appointment above a particular price
Actor: Patient
Precondition:
Steps:
Actor action: The patient views the appointment above a particular price

System Responses: the list of appointments above a price is displayed
Post Condition: system displays the list of appointments for the condition

5. **Use Case:** View the appointments made by a patient
Description: The patient views the appointments made by him/her
Actor: Patient
Precondition: The patient must have made at least one appointment to view an appointment
Steps:
Actor action: The patient views the history of the appointment
System Responses: Displays all the appointments made by a patient
Alternate Path: There are no appointments made by a patient
Error: No history of appointments available.

6. **Use Case:** Cancel the appointments made by a patient
Description: The patient can cancel the appointments made by him/her
Actor: Patient
Precondition: The patient must have made at least one appointment to cancel an appointment
Steps:
Actor action: The patient can view cancel history of the appointment
System Responses: Displays all the cancel appointments made by a patient
Alternate Path: There are no appointments cancelled by a patient
Error: No history of cancelled appointments available.

7. **Use Case:** Reschedule the appointments made by a patient
Description: The patient can reschedule the appointments made by him/her
Actor: Patient
Precondition: The patient must have made at least one appointment to reschedule an appointment
Steps:
Actor action: The patient can view reschedule history of the appointment
System Responses: Displays all the rescheduled appointments made by a patient
Alternate Path: There are no appointments rescheduled by a patient
Error: No history of rescheduled appointments available.

8. **Use Case:** Doctor can edit the appointments made by a patient
Description: The doctor can edit the appointments made by him/her
Actor: Doctor
Precondition: The doctor must have made at least one appointment to edit an appointment
Steps:
Actor action: The doctor can view edit history of the appointment
System Responses: Displays all the edited appointments made by a patient

Alternate Path: There are no appointments edited by a doctor
Error: No history of edited appointments available.

9. **Use Case:** View the payment details made by the patient
Description: The patient can view the payment details made by him/her
Actor: Patient
Precondition: The patient must have made at least one payment details to view an appointment
Steps:
Actor action: The patient can view payment details of the appointment
System Responses: Displays all the payment details made by a patient
Alternate Path: There are no payment details by a patient
Error: No history of payment details available.

10. **Use Case:** Patient can find a doctor based on his/her specialization
Description: The patient can book an appointment on the basis of doctor specialization
Actor: Patient
Precondition: The patient must book appointment based of his/her health issues
Steps:
Actor action: The patient can view specialized doctors
System Responses: Displays all the specialized doctor's for a patient
Alternate Path: There are no specialized doctor's
Error: No history of specialized doctor's available.

RELATIONAL-ALGEBRA EXPRESSIONS FOR THE USE CASES

1.Use Case: View an appointment already ordered through Twitter

Π{w.appointment_url}(σ{w.appointment_id = t.appointment_id ∧ t.Twitter_handle = '@anna' }(ρ{w}(Women_Appointment) × ρ{t}(Twitter_Order)))

2.Use Case: View the products above a particular price (say $100)

Π{w.appointment_url, w.appointment_url}(σ{w.price > 100 }(Women_Appointment))

3.Use Case: View the orders made by a user

Π{s.Twitter_handle, s.appointment_id}(σ{s.Twitter_handle = '@emma' }(Payment))

SQL STATEMENTS

1. Use Case: Register for an appointment at Brigham and Women's Hospital

```
INSERT INTO Hospital_Account
(Twitter_handle, login, password)
VALUES (@anna, anna123, xxxxxx)

INSERT INTO Hospital_Account
(Twitter_handle, login, password)
VALUES (@emily, emily123, xxxxxx);

INSERT INTO Hospital_Account
(Twitter_handle, login, password)
VALUES (@emma, emma123, xxxxxx);
```

2. Use Case: Make an appointment at Brigham and Women's Hospital

```
INSERT INTO Tweets
(tweet_id, Twitter_handle, tweet_text, profile_image_url, created_at )
VALUES (12321, @anna, 'I would like to book  appointment
https://www.brighamandwomens.org//product_id=2449' , 'www.facebook.com/emma.smith/
photo.php?fbid=10205' , 12-11-2022  );

INSERT INTO Tweet_url
(tweet_id, hospital_url )
VALUES (12321,'https://www.brighamandwomens.org//product_id=2449');

INSERT INTO Twitter_Appointment
(appointment_ID, patient_ID, doctor_ID, appointmet_date, appointment_reason,
twitter_handle)
VALUES (4532, 12321, @john, 2341, ,13/11/2022, Fever, emma123 )

INSERT INTO Payment
(payment_id, payment_date, total_amount, payment_details, appoinment_ID
Twitter_handle)
VALUES ( 9876, 13/11/2022, 1, $78.4,2453, @emma123 )
```

3. Use Case: View an appointment already ordered through Twitter

```
SELECT w.appointment_url
FROM Women_Appointment w, Twitter_Appointment t
WHERE
```

t.appointment_id = w.appointment_id AND
t.Twitter_handle = '@emaa123'

4. Use Case: View the appointments above a particular price (say $100)

```
SELECT w.patient_name, w.appointment_url
FROM Women_Appointments w
WHERE
w.price > 100;
```

5. Use Case: View the appointments made by a patient

```
        SELECT s.Twitter_handle, s.patient_id
FROM Payment s
WHERE
s.Twitter_handle = 'anna';
```

SQL Statements for the conceptual model:

User Table:

```
CREATE TABLE `User` (
  `Twitter_handle` VARCHAR(10),
  `name` VARCHAR(20),
  `profile_image_url` VARCHAR(200),
  `description` VARCHAR(100),
  `followers_count` INT,
  `following_count` INT,
  PRIMARY KEY  (`Twitter_handle`)
);
```

Tweets Table:

```
CREATE TABLE `Tweets` (
  `tweet_id` INT NOT NULL AUTO_INCREMENT,
  `Twitter_handle` VARCHAR(10),
  `tweet_text` VARCHAR(140),
  `profile_image_url` VARCHAR(200),
  `created_at` DATETIME,
  PRIMARY KEY  (`tweet_id`)
);
```

Tweet_Tags Table:

```
CREATE TABLE `Tweet_Tags` (
  `tweet_id` INT NOT NULL,
  `tags` VARCHAR(20),
```

```
  PRIMARY KEY  (`tweet_id`)
);
```

Tweet_Tags Table:

```
CREATE TABLE `Tweet_Tags` (
  `tweet_id` INT NOT NULL,
  `tags` VARCHAR(20),
  PRIMARY KEY  (`tweet_id`)
);
```

Tweet_Mentions Table:

```
CREATE TABLE `Tweet_Mentions` (
  `tweet_id` INT NOT NULL,
  `source_user` VARCHAR(10),
  `target_user` VARCHAR(10),
  PRIMARY KEY  (`tweet_id`)
);
```

Tweet_Url Table:

```
CREATE TABLE `Tweet_Url` (
  `tweet_id` INT NOT NULL,
  ‘hospital_url’ VARCHAR(200),
  PRIMARY KEY  (`tweet_id`)
);
```

Hospital_Account Table:

```
CREATE TABLE `Hospital_Account` (
  `Twitter_handle` VARCHAR(10) NOT NULL,
  `password` VARCHAR(10),
  ‘login’ VARCHAR(10),
  PRIMARY KEY  (`Twitter_handle`)
);
```

Twitter_Appoinment Table:

```
CREATE TABLE `Twitter_Appointment` (
  `appointment_id` INT NOT NULL AUTO_INCREMENT,
  `doctor_id` INT NOT NULL,
  `patient_id` INT NOT NULL,
  `appointment_date` INT NOT NULL,
  ‘appointment_reason’ VARCHAR(10),
```

```
  `Twitter_handle` VARCHAR(10) NOT NULL,
  PRIMARY KEY  (`appointmentt_id`)
);
```

Patient Table:

```
CREATE TABLE `Patient` (
  `patient_id` INT NOT NULL AUTO_INCREMENT,
  `department_schedule` VARCHAR,
  `product_id` INT,
  ‘name’ VARCHAR(20),
  ‘phone’ INT,
  ‘email’ VARCHAR(20),
  `appointment_id` INT,
  PRIMARY KEY  (`patient_id`)
);
```

Payment Table:

```
CREATE TABLE `Payment` (
  `payment_id` INT NOT NULL AUTO_INCREMENT,
  `total_amount` FLOAT,
  `Twitter_handle` VARCHAR(10),
  `pay_details` INT,
  `pay_date` INT,
  PRIMARY KEY  (`payment_id`)
);
```

Doctor Table:

```
CREATE TABLE `Doctor` (
  `doctor_id` INT NOT NULL AUTO_INCREMENT,
  `speciality` VARCHAR(20),
  `Twitter_handle` VARCHAR(10),
  `location` VARCHAR(20),
  `language` VARCHAR(10),
   `gender` VARCHAR(10),
  PRIMARY KEY  (`doctor_id`)
);
```